Dipartimento di Informatica

Università del Piemonte Orientale "A. Avogadro"

Via Bellini 25/G, 15100 Alessandria

http://www.di.unipmn.it

Non deterministic Repairable Fault Trees for computing optimal repair strategy

Authors: *Marco Beccuti (beccuti@mfn.unipmn.it),*

Giuliana Franceschinis (giuliana@mfn.unipmn.it),

Daniele Codetta-Raiteri (raiteri@mfn.unipmn.it),

Serge Haddad (haddad@lsv.ens-cachan.fr).

TECHNICAL REPORT TR-INF-2008-07-05-UNIPMN

(July 2008)

The University of Piemonte Orientale Department of Computer Science Research Technical Reports are available via WWW at URL `http://www.di.unipmn.it/`.

Plain-text abstracts organized by year are available in the directory

Recent Titles from the TR-INF-UNIPMN Technical Report Series

2008-04 *Reliability and QoS Analysis of the Italian GARR network*, Bobbio, A., Terruggia, R., June 2008.

2008-03 *Mean Field Methods in performance analysis*, Gribaudo, M., Telek, M., Bobbio, A., March 2008.

2008-02 *Move-to-Front, Distance Coding, and Inversion Frequencies Revisited*, Gagie, T., Manzini, G., March 2008.

2008-01 *Space-Conscious Data Indexing and Compression in a Streaming Model*, Ferragina, P., Gagie, T., Manzini, G., February 2008.

2007-05 *Scheduling Algorithms for Multiple Bag-of-Task Applications on Desktop Grids: a Knowledge-Free Approach*, Canonico, M., Anglano, C., December 2007.

2007-04 *Verifying the Conformance of Agents with Multiparty Protocols*, Giordano, L., Martelli, A., November 2007.

2007-03 *A fuzzy approach to similarity in Case-Based Reasoning suitable to SQL implementation*, Portinale, L., Montani, S., October 2007.

2007-02 *Space-conscious compression*, Gagie, T., Manzini, G., June 2007.

2007-01 *Markov Decision Petri Net and Markov Decision Well-formed Net Formalisms*, Beccuti, M., Franceschinis, G., Haddad, S., February 2007.

2006-03 *New challenges in network reliability analysis*, Bobbio, A., Ferraris, C., Terruggia, R., November 2006.

2006-03 *The Engineering of a Compression Boosting Library: Theory vs Practice in BWT compression*, Ferragina, P., Giancarlo, R., Manzini, G., June 2006.

2006-02 *A Case-Based Architecture for Temporal Abstraction Configuration and Processing*, Portinale, L., Montani, S., Bottrighi, A., Leonardi, G., Juarez, J., May 2006.

2006-01 *The Draw-Net Modeling System: a framework for the design and the solution of single-formalism and multi-formalism models*, Gribaudo, M., Codetta-Raiteri, D., Franceschinis, G., January 2006.

2005-06 *Compressing and Searching XML Data Via Two Zips*, Ferragina, P., Luccio, F., Manzini, G., Muthukrishnan, S., December 2005.

2005-05 *Policy Based Anonymous Channel*, Egidi, L., Porcelli, G., November 2005.

2005-04 *An Audio-Video Summarization Scheme Based on Audio and Video Analysis*, Furini, M., Ghini, V., October 2005.

Contents

1 Introduction **2**

2 Related work **4**

2.1 Fault Trees 4

2.2 Tools for FT analysis 4

2.3 Repairable Fault Trees 5

3 Non deterministic RFT **6**

3.1 NdRFT syntax 6

3.2 MDP semantics of NdRFT 8

3.3 Discussion 11

4 Translation from NdRFT to MDPN **12**

4.1 Generating the PR subnet 13

4.2 Generation of the ND subnet 16

4.3 Translation correctness 19

4.4 An transition priority assignment improving efficiency 23

5 Framework architecture **24**

6 Experiment results **25**

7 Conclusion and future work **28**

A Markov Decision Petri Net **29**

A.1 MDPN semantics. 31

Non deterministic Repairable Fault Trees for computing optimal repair strategy

Marco Beccuti, Giuliana Franceschinis, Daniele Codetta-Raiteri
Dip. di Informatica, Univ. del Piemonte Orientale
Via Bellini, 25/G
15100 Alessandria, Italy
{beccuti, giuliana, raiteri}@mfn.unipmn.it

Serge Haddad
LSV, ENS Cachan, CNRS
61, avenue du Président Wilson
Cachan, France
haddad@lsv.ens-cachan.fr

Abstract

In this paper, the *Non deterministic Repairable Fault Tree* (NdRFT) formalism is proposed: it allows to model failure modes of complex systems as well as their repair processes. The originality of this formalism with respect to other Fault Tree extensions is that it allows to face repair strategies optimization problems: in an NdRFT model, the decision on whether to start or not a given repair action is non deterministic, so that all the possibilities are left open. The formalism is rather powerful allowing to specify which failure events are observable, whether local repair or global repair can be applied, and the resources needed to start a repair action. The optimal repair strategy can then be computed by solving an optimization problem on a *Markov Decision Process* (MDP) derived from the NdRFT. A software framework is proposed in order to perform in automatic way the derivation of an MDP from a NdRFT model, and to deal with the solution of the MDP.

Keywords. Fault Tree, Optimal repair strategy, Markov Decision Process, Markov Decision Petri Net

1 Introduction

The Fault Trees (FT) [22] are a well-known formalism for the evaluation of dependability of complex systems. They provide an intuitive representation of the system in terms of its faults, modeling how the combinations of failure events relative to the components of the system, can cause the failure of the sub-systems or of the whole system.

Many extensions of this formalism have been proposed in order to enhance the advantages of the FT for the design and the assessment of the systems (e.g. Dynamic FT [16], Parametric FT [5], etc.). Among these extensions, in [12] the Repairable FT (RFT) was presented in order to evaluate the effect of different repair policies on a repairable system.

In this paper, we present a new FT extension, called *Non deterministic Repairable Fault Tree* (NdRFT) which has been designed to define and solve repair strategy optimization problems: in an NdRFT model the possible repair strategies are not predefined; on the contrary, the best strategy, minimizing the failure probability of the global system, is automatically computed. This is done by defining the NdRFT semantics in terms of a *Markov Decision Process* (MDP), a formalism embedding non deterministic and probabilistic behavior [14, 18], and then solving the optimization problem using the methods available for MDPs.

The generation of the MDP is achieved by an intermediate translation of the NdRFT model into a *Markov Decision Petri Net* (MDPN) [3]: this allows to reuse the efficient algorithms devised to derive an MDP from an MDPN. Moreover a direct translation from NdRFT to MDP requires to implement a mechanism to combine the failure/repair events of all components into a single complex transition or action: this is already implemented for MDPN formalism.

The NdRFT formalism allows to express in an elegant way several possible start repair options based on: 1) the concept of "observability" of events (repair actions can only be triggered by observable failures), 2) the notion of local versus global repair action, 3) the notion of repair supervisor component, in case of global repair. Very few restrictions are imposed on the scope of repair actions (so that the repair of each basic component can start based on observations made on different failure events). The NdRFT formalism allows the modeler to express in a familiar language (NdRFT extends FT) the failure mode and the repair options in the system; in this way, he avoids to deal with a larger, unstructured and state-level MDP model that is instead derived from the NdRFT model.

The paper is structured as follows: Sec. 2 presents some related work about FT, tools for FT analysis, and RFT; in Sec. 3 we provide the formal definition of the NdRFT formalism; Sec. 4 explains how to derive from a NdRFT model, the corresponding MDPN; in Sec. 5 we present a software framework for the design and the solution of NdRFT models in order to compute the optimal repair strategy and the corresponding dependability of the system; finally in section 6, we present and analyze some experimentations.

2 Related work

2.1 Fault Trees

In the FT formalism, nodes can belong to one of these two categories: events and gates. Events concern the failure of components, subsystems or of the whole system. We can consider an event as a Boolean variable: it is initially $false$ and it becomes $true$ when the failure occurs.

An example is shown in Fig. 1.b. The events graphically represented as a rectangle with an attached circle are called *Basic Events* (BEs) and model the failure of the components of the system; such events are stochastic, so their occurrence is ruled by some probability distribution.

The events depicted simply by a rectangle represent the failure of subsystems; we call them *Internal Events* (IEs) and they are the output of a gate node. *Gates* are connected by means of arcs to several input events and to a unique output event; the effect of a gate is the propagation of the failure to its output event if a particular combination of its input events occurs. In the standard version of the FT formalism three types of gate are present and correspond to the AND, OR and *"K out of N"* Boolean functions.

Finally, we have a unique event called *Top Event* (TE), modeling the failure of the whole system. The FT incorporates a Boolean formula expressing the TE truth value as a function of its variables (BEs).

The analysis of an FT model returns several dependability measures such as the system reliability, the system minimal cut-sets, the criticality of each component [22]; in particular, the system *reliability* at time t is the probability that the system has been working in the time interval $(0, t)$. The most efficient way to perform the analysis of an FT, consists of generating the *Binary Decision Diagram* (BDD) [7] representing the same Boolean formula expressed by the FT: efficient algorithms allow to compute on the BDD the measures cited above [19].

2.2 Tools for FT analysis

Several software tools support the FT analysis. Some of them can deal with the repair, but they allow only to model the repair of single components: the repair process is triggered by the component failure and has effect only on the same component. For instance in the tool ASTRA [13] developed by the *European Commission Joint Research Centre* (JRC), one of the parameters of a BE is the time to repair the component whose failure is modeled by the same BE. In other tools, the time to repair a component is a random variable ruled by some distribution such as the negative exponential one. This is the case of the following tools where a repair rate can be associated with a BE: *Stars Studio* developed by JRC [13], *HIMAP* [15] by Iowa State University, *Relex* [26], *FTA-Pro* [24] by Dyadem, *FaultTree+* [27] by Isograph Software, *FTAnalyzer* [28] by Advanced Logistic Development (ALD).

The *SHARPE* tool [21] allows hierarchical modeling: the probability to occur of a BE can be set equal to some measure computed on another kind of model, for instance a *Continuous Time Markov Chain* (CTMC) [21]. In this way, the failure and repair mode of a component may be more complex than a simple transition from the working state to the failure state and vice-versa. In any case, the model representing the failure and repair mode of the component has to be manually drawn by the modeler. Hierarchical modeling is possible by means of the *HIMAP* tool as well [15].

A *Dynamic Fault Trees* (DFT) [1] is a particular extension of FT where dependencies between BEs can be set by means of the *dynamic gates*. The analysis of a DFT model can be performed by conversion into a *Continuous Time Markov Chain* (CTMC) [16] and is supported by the tool *Galileo* [25]. In [6], a DFT model can include the repair of components, and the analysis of the model is faced in by exploiting *Input-Output Interactive Markov Chains*, however each repair action still concerns a single component.

2.3 Repairable Fault Trees

In the literature, the *Repairable Fault Tree* (RFT) formalism [12] is the only extension of FT that allows to model the repair of a subsystem when triggered by a specific failure event. This means that the repair process concerns a set of components instead of a single one. Moreover, in the RFT formalism, the repair action is not simply ruled by a repair rate, but it is influenced by a *repair policy*: defining a repair policy in a RFT model means setting the parameters ruling each aspect of the repair process, such as the mean time to detect the failure, the mean time to repair a single component or a set of components, the number of repair facilities, the order of repair of the components. From a RFT model we can compute the system *availability* at time t; this means the probability that the system is working at time t.

The RFT differs from the FT, for the introduction of a new primitive called *Repair Box* (RB) [12] allowing the model designer to represent the presence of a repair process involving a certain set of components called *basic coverage set* (Cov_{BE}) of the RB; such action is activated by the occurrence of a specific failure event called *trigger event* and concerning a component or a subsystem. The effect of the RB is setting the value of the BEs in its Cov_{BE} to *true* (working), if their current value is $false$ (failed). Such repair action is performed according to the repair policy associated with the RB node. Actually, the effect of the RB does not influence only the BEs in its basic coverage set, but also all the IEs whose value can be expressed by a Boolean function over a set of BEs including at least one BE in Cov_{BE}. In [12], the computation of the system *availability* from its RFT model, has been faced by converting the RFT model into a *Generalized Stochastic Petri Net* (GSPN).

In the RFT formalism, the repair policy (or strategy) is defined by the modeler and is associated with the RB primitive; therefore the only way for the modeler to identify the best policy, consists of analyzing

the system according to several repair policies by constructing several RFT models, and by comparing the system availability values returned by the RFT models analysis. So, the RFT formalism does not allow to automatically determine the best repair policy.

The possibility to determine the optimal repair policy given all the repair possibilities, is an issue concerning several fields of engineering. So far, this problem has been usually faced in the literature in analytical ways, typically in form of operative research problems [8, 23, 20]. The NdRFT formalism presented in this paper, is an attempt to deal with the problem of optimal repair strategy, by building a graph based model having an intuitive notation and allowing to model several repair options together with the failure combinations in the system.

3 Non deterministic RFT

3.1 NdRFT syntax

In this section the formal definition of the NdRFT is provided and commented through an example.

Definition 1 (Non deterministic Repairable FT) *An NdRFT is a five-tuple:*

$$\mathcal{S} = \langle \mathcal{E}, \mathcal{G}, \mathcal{A}, \mathcal{R}, res_0 \rangle$$

where:

$\mathcal{E}$ *is the set of events.*

$\mathcal{G}$ *is the set of gates;* $\mathcal{E} \cap \mathcal{G} = \emptyset$. *A gate* g *has a type*[1] *denoted* $g.type \in \{\texttt{and}, \texttt{or}\}$.

$\mathcal{A}$ *is the set of arcs, a subset of* $\mathcal{E} \times \mathcal{G} \cup \mathcal{G} \times \mathcal{E}$. *For* x *belonging to* $\mathcal{E} \cup \mathcal{G}$, *we denote* $x^\bullet \equiv \{y \mid (x, y) \in \mathcal{A}\}$ *and* ${}^\bullet x \equiv \{y \mid (y, x) \in \mathcal{A}\}$. $\mathcal{A}$ *satisfies:*

1. $\forall g \in \mathcal{G}, |g^\bullet| = 1$ *and* $\forall e \in \mathcal{E}, |{}^\bullet e| \leq 1$
2. *There is exactly one event, denoted* $\top$ *and called Top Event, s.t.* $\top^\bullet = \emptyset$; *all other events satisfy* $|e^\bullet| \geq 1$
3. *The set of events can be partitioned into* basic *events* $\underline{\mathcal{E}} \equiv \{e \mid {}^\bullet e = \emptyset\}$ *and* internal *events* $\overline{\mathcal{E}} \equiv \{e \mid {}^\bullet e \neq \emptyset\}$
4. *The (directed) graph induced by* $\mathcal{A}$ *is acyclic.*

$\mathcal{R}$ *is a finite set of repair resource types;* $res_0 \in Bag(\mathcal{R})$ *is the multiset of available resources, where* $Bag(\mathcal{R})$ *is a generalization of a set, so that it can contain several occurrences of the same element.*

Each event is associated with a set of attributes, related to its failure probability and to the definition of the

[1] Since the proposed optimization method is based on the state space, other gate types could easily be considered, including dynamic ones: in this paper only and/or gates are considered for the sake of space.

applicable repair actions. Any event e is either observable *(e.obs* = `true`*) or* non observable *(e.obs* = `false`*); only observable events can trigger a repair action.*

Moreover, each BE e has the following additional attributes:

1. a fault probability denoted e.fprob ranging over $[0, 1]$*;*

2. a repair attribute denoted e.rep $\in \{\texttt{true}, \texttt{false}\}$ *indicating if the event is repairable or not; if e.rep* = `true`*, e has also a repair probability denoted e.rprob* $\in [0, 1]$ *and a multiset of required resources denoted e.res* $\in Bag(\mathcal{R})$*.*

Finally, each internal observable event e has the following additional attributes:

1. a set of BEs that should be repaired in case of e failure, denoted e.torep such that $e' \in e.torep \Rightarrow e'.rep = true$*, moreover there is a path from e to* e' *according to* $\mathcal{A}$*;*

2. a repair strategy denoted e.str $\in \{\texttt{global}, \texttt{local}\}$*. When the strategy associated with an event is global, it also has a repair probability denoted e.rprob* $\in [0, 1]$ *and a multiset of required resources denoted e.res* $\in Bag(\mathcal{R})$*.*

Let us comment the above definition by means of the example of Fig. 1 (whose meaning will be explained in Sec. 6): in the picture the events are depicted in a different way according to their *obs* and *rep* attribute values. Down arrows, labeled with a number, next to BEs indicate their failure probabilities; up arrows, labeled with a number, next to repairable BEs or to internal events with global strategy, indicate the repair probability. Basic events $A3$ and $P3$ in the example are not repairable. In the NdRFT formalism, the assumption of discrete time holds: the time to fail (repair) a component is ruled by the geometric distribution having as parameter the failure (repair) probability (see section 3.3).

The failure of an observable and repairable BE e (e.g. $A1$) can immediately trigger a repair action of the component, while the repair of a non observable (but repairable) event e (e.g. $A2$) can only be triggered by an observable internal event connected to e (for $A2$ it can be $U2$ or TE). Intuitively, observability is related to the possibility of detecting a failure. In the example of Fig. 1 we have only one type of resource and each repair action requires only one resource (observe that any local repair action, including the one triggered by the TE, requires one resource for *each* BE to be repaired).

The event attribute *repair strategy* defines the granularity of the repair process triggered by the occurrence of the (internal) event e: if the repair strategy is *global* (as for $U2$ in the example), *all* the repairable basic components in *e.torep* ($A2$ and $P2$ in the example) are repaired simultaneously and brought back to the working state when the global repair process terminates. This means that a global repair process is a unique repair process (e.g. representing the substitution of a down server with a new server: all components are substituted at once); while a global repair action is ongoing, the basic components in *e.torep* cannot be simultaneously involved in any other repair action (global or local). If instead the repair policy is *local* (as

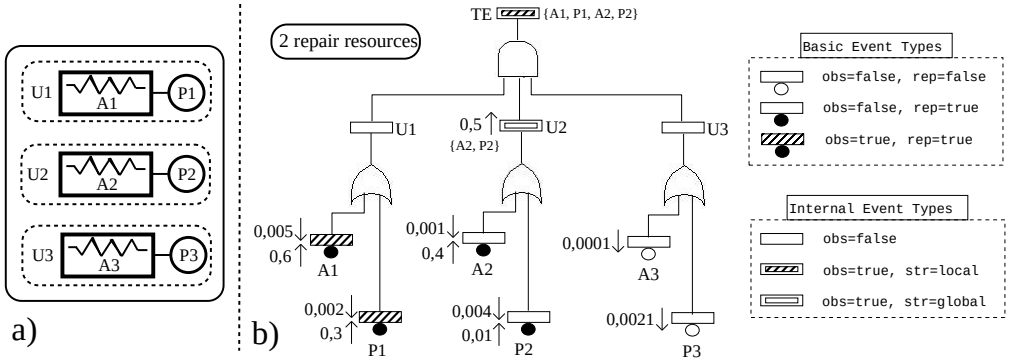

Figure 1: a) The block scheme of the AHRS. b) The NdRFT model of the AHRS.

for TE in the example), for each repairable BE component in *e.torep*, it is possible to decide to repair or not such component; moreover the single components repair may not start simultaneously (e.g. because there are not enough resources). A BE can appear in the *torep* set of several internal events; for example $A2$ and $P2$ are in the *torep* set of both $U2$ and TE: when a failure has occurred for only one among the two BEs, the local strategy could be more appropriate, but it can be activated only if TE has occurred already. Otherwise, if both $A2$ and $P2$ failure has occurred, the global repair of $U2$ may be more convenient. Observe that given the example NdRFT structure, $U2$ can immediately witness the failure of one or both events $A2$ and $P2$, and trigger the substitution of both.

3.2 MDP semantics of NdRFT

MDP definition. A (discrete time and finite) MDP is a dynamic system where the transitions between states follow a two-step process. First, one non deterministically selects an action inside the subset of enabled actions. Then one samples the new state with respect to a probability distribution depending on the current state and the selected action. The non deterministic step represents a decision taken by a controller in order to manage the system, or a behavior triggered by the environment that the system cannot control. Our approach is based on the former interpretation. The probabilistic step takes into account that the effect of an action statistically depends on non modeled (or unknown) parameters.

In order to formally define the objective to optimize, one associates a reward with any state and selected action (the reward can also be interpreted as a cost). The following definition formalizes these concepts.

Definition 2 (Markov Decision Process, MDP:) *An MDP $\mathcal{M}$ is a four-tuple $\mathcal{M} = \langle S, A, p, r\rangle$ where:*

1. *S is a finite set of states,*
2. *A is a finite set of actions defined as $\bigcup_{s\in S} A_s$ where A_s is the set of enabled actions in state s,*

3. $\forall s \in S, \forall a \in A_s, p(\cdot|s,a)$ *is a (transition) probability distribution over* S *such that* $p(s'|s,a)$ *is the probability to reach* s' *from* s *by triggering action* a*,*

4. $\forall s \in S, \forall a \in A_s, r(s,a) \in \mathbb{R}$ *is the reward associated with state* s *and action* a*.*

Once an action choice is fixed, the MDP behaves like a Markov chain and different global measures on the random path can be defined as for example the (discounted) sum of rewards or the average of the rewards. The goal of the analysis is computing the optimal value of the measure, and when possible, computing the associated strategy. In finite MDPs, efficient solution techniques have been developed to this purpose [18] and different tools are based on this theory (see for instance the experiment section).

NdRFT semantics. The semantics of an FT is simply a Boolean formula expressing the TE truth value as a function of the BEs truth value; the possible (minimal) failure configuration leading to the TE and their occurrence probability (at time t) can be efficiently computed using a BDD [19] representation of the FT, without need to develop its dynamic failure behavior. NdRFT semantics instead (as well as RFT one) requires to explicitly expand and analyze the dynamic behavior of the model since the introduction of the repair processes adds the possibility for events to switch between the up and down state several times within a given observation period.

In this paragraph, we will define precisely the dynamic behavior of an NdRFT, which can be described by an MDP. Let us first define the MDP states:

Definition 3 (MDP$_{NdRFT}$ state) *A state* ρ *of the MDP corresponding to a given NdRFT is a tuple:*

$$\rho = \langle \{st_e\}_{e \in \mathcal{E}}, \{sup_e\}_{e \in \mathcal{E}} \rangle$$

where $st_e = \{Up, Down, LocRep, GlobRep_u, GlobRep_d\}$ *represents the state of the component/subsystem whose failure corresponds to the event* e*. If* $st_e \in \{Up, GlobRep_u\}$*, then* $e = false$ *in the fault tree, if* $st_e = \{Up, LocRep, GlobRep_d\}$*, then* $e = true$ *in the fault tree. Only BEs can be in repair state, and for these events* sup_e *represents the supervisor of the repair process: this is the basic event* e *itself if the repair action is local, while in case of global repair the supervisor is the internal event that triggered it. Observe that* $st_e \in \{Up, Down\} \Leftrightarrow sup_e = NULL$*. The state of the internal events in* ρ *can only be in* $\{Up, Down\}$ *and it can be derived from the state of the BEs and the FT structure. The initial state* ρ_0 *is:* $\rho_0 : \forall e \in \mathcal{E}, st_e^0 = Up \wedge sup_e = NULL$*.*

Let us define the set A_ρ of actions that can be chosen in state ρ: each action $a \in A_\rho$ is a mapping $\underline{\mathcal{E}} \cup \overline{\mathcal{E}}_{GR} \rightarrow \{repair, not\_repair\}$; in other words an action is a set of decisions on whether a local or global repair process should start for a given component/subsystem. An action a can be taken in ρ if (1) the components or subsystems for which a repair action is required are not working (i.e. they are down) and (2)

there are enough resources to perform all the scheduled repair actions (both those already ongoing in ρ and the new ones just started as specified by a).

For each state ρ, it is possible to define the multiset res_ρ of busy resources as: $res_\rho = \sum_{e \in sup(\mathcal{E})} e.res$. Of course at each time the following condition must be verified: $res_\rho \subseteq res_0$ that can also be expressed as $\forall r \in Bag(\mathcal{R}), res_\rho(r) \leq res_0(r)$, where $res_i(r)$ denotes the multiplicity of r in res_i.

Once an admissible action $a \in A_\rho$ is chosen, an intermediate state $\langle \rho, a \rangle$ is reached: here a probability distribution allows to determine the state change; the probability distribution can be derived from the probability distribution of failure and repair completion events, as detailed hereafter.

Summarizing, the dynamic of the MDP corresponding to an NdRFT, is defined in terms of two steps: a non deterministic one (selecting the subset of repair actions that should start) and a probabilistic one (probabilistically choosing the newly occurred failure events and which among the ongoing repair actions have completed).

The state change induced by each step is defined as follows:

Non Deterministic step: MDP actions. This step comprises a (possibly empty) set of repair start decisions for BEs and intermediate events triggering a global repair. Each repair must be triggered by (basic or internal) *observable* events that are in state *Down*. For each repair start decision, the supervisor of the involved event must be specified: it is the event itself in case of local repair, while it is the internal trigger event for global repair.

The conditions for the two types of state change are:

$st_e : Down \rightarrow LocRep$: (1) $e.obs = true \ \wedge \ e.rep = true$ or (2) $\exists e' : e \in e'.torep, st_{e'} = Down, e'.obs = true, e'.str = local$; in both cases $sup_e = e$.

$st_e : Down \rightarrow GlobRep_d$ or $st_e : Up \rightarrow GlobRep_u$: $\exists e' : e \in e'.torep, st_{e'} = Down, e'.obs = true, e'.str = global$; in this case the state change must happen simultaneously for every $e \in e'.torep$ and then $sup_e = e'$.

The set of repair processes chosen to start, defining an action a, lead to the new state $\langle \rho, a \rangle$: of course state $\langle \rho, a \rangle$ must be consistent with the requirement $res_{\langle \rho, a \rangle} \subseteq res_0$.

The possible actions A_ρ available in state ρ of the MDP are thus all the legal repair start decision sets satisfying the conditions and the resource constraints described above.

Probabilistic step. In this step the possible state changes for each BE e are:

$st_e : Up \rightarrow Down$: with probability $e.fprob$ (or remain in the Up state with probability $1 - e.fprob$);

$st_e : LocRep \rightarrow Up$: with probability $1 - e.rprob$ (or remain in the $LocRep$ state with probability $e.rprob$);

$st_e : GlobRep_* \rightarrow Up$: with probability $1 - sup(e).rprob$ (or remain in the $GlobRep_*$ state with probability $sup(e).rprob$): this state change must happen simultaneously for all $e'' \in sup(e).torep$ (it is a *single probabilistic choice* with synchronous effect on all events involved in the repair action). For any event e that returns to the Up state, sup_e is reset to $NULL$. Hence the probabilistic choice that follows a given non deterministic

action in the MDP, leading from the intermediate state $\langle \rho, a \rangle$ to state ρ' corresponds to a probabilistic step as described above: the probability of each step is obtained as the product of the probabilities of each single event state transition (we recall again that the end of a global repair represents a single event, independently on how many basic events are involved).

As already remarked above, the states of the internal events are derived from those of the BEs using the FT structure.

This completes the definition of the MDP underlying a given NdRFT. The optimization problem has the following goal: minimizing the probability (at time t or in steady state) of being in a state where the TE failure has occurred. Different goals might be defined as well, e.g. taking into account the cost of repair actions or the cost of having a system working in a degraded mode.

In practice, the computation of the optimal strategy requires three steps: (1) generation of the MDP from the NdRFT, (2) analysis of the MDP, (3) presentation of the results in a form that is understandable for the designer.

These steps can be automatized. The first step can be implemented in two ways: defining an algorithm that generates the set of reachable states, the corresponding non deterministic actions and consequent probabilistic state change, or translating the NdRFT in an intermediate model for which the above tasks have already been defined and implemented. In this paper we propose to use the second approach and provide an algorithm for translating an NdRFT into a *Markov Decision Petri Net* (MDPN) [3]. From the MDPN model an MDP can be automatically derived.

3.3 Discussion

The NdRFT model is a discrete time one. This can be justified by the fact that faults in plants are often detected at the time a sampling is performed through some sensor: sampling is usually done periodically according to a synchronous schema. Due to the discrete time assumption, the specification of the failure and repair process of each (basic) *repairable* component x is given by probability $P_{Failure}(x)$ and $P_{Repair}(x)$. $P_{Failure}(x)$ (resp. $P_{Repair}(x)$) represents the probability that a failure (resp. the end of the repair) occurs at any (discrete) time step provided the corresponding component is up (resp. is down and under repair). As a consequence, the time to failure of a component, and its repair time have geometric distribution:

$P(TtF_e = k) = (1 - P_{Failure}(e))^{k-1} P_{Failure}(e)$

$P(repTime_e = k) = (1 - P_{Repair}(e))^{k-1} P_{repair}(e)$

In NdRFT the repair policy is not completely specified (instead, this is the case for RFT): the choice to repair or not a repairable components fault is non deterministic. This leads to an MDP semantics: as

a consequence, we can compute the optimal repair strategy minimizing the failure probability of the global system. Observe that even without taking into account the cost of repair, finding the optimal strategy is not trivial, since we account for limited repair resources (each repair action can be associated with a multiset of required resources to complete it).

In the NdRFT we can model processes where the components or the subsystems under repair return available as soon as possible (maybe in a degraded state) without waiting the repair of all its down BE components. Moreover, the notion of observability allows to specify when a fault can be detected, and hence when the corresponding repair activity can start (this generalizes the notion of *trigger event*).

Finally repair actions may involve common components: this choice increases the flexibility in the choice among the possible repair strategies that may be pursued, still allowing a simple and clean semantics based on the notions of observability and of global vs. local repair strategy.

4 Translation from NdRFT to MDPN

In this section we are going to describe how to obtain from an NdRFT model the corresponding MDPN model. An informal introduction to the MDPN formalism is provided first, then the pattern-based translation algorithm is presented.

The generation of the MDP from the MDPN model can be performed as described in [3]. The MDP obtained is solved in order to find the optimal repair strategy (at finite horizon t or in steady state, as appropriate) and the corresponding Top Event failure probability (or any other dependability measure).

A brief introduction to MDPNs. MDPNs were first introduced in [3] as high level models to specify the behavior of an MDP. The main features of the high level formalism are the possibility to specify the general behavior as a composition of the behavior of several components (some of which are controllable [2]); moreover each MDP non deterministic or probabilistic transition can be composed by a set of non deterministic or probabilistic steps, each one involving a subset of components.

An MDPN model is composed of two parts, both specified using the PN formalism with priorities associated with transitions: the PN^{nd} subnet and the PN^{pr} subnet (describing the non deterministic (ND) and probabilistic (PR) behavior respectively). The two subnets share the set of places, while having disjoint transition sets. In both subnets the transitions are partitioned into "run" and "stop" subsets, and each transition has an associated set of components involved in its firing (in the PN^{nd} only controllable components can be involved). Transitions in PN^{pr} have a "weight" attribute, used to compute the probability of each firing sequence. Run transition firings represent intermediate steps in a ND/PR transition at the MDP level, while

[2] A component that is subject to local non deterministic choice will be called controllable component, otherwise it will be called non controllable component.

Stop transitions represent the final step in a ND/PR transition, for all components involved in it. An MDPN model behavior alternates between ND transition sequences and PR transition sequences, initially starting from a ND state. The PR sequences are determined according to the PN^{pr} structure, start with a PR state reached by a ND state, and include exactly one stop transition for each component; the ND sequences are determined by the PN^{nd} structure, start from a ND state reached by a PR state, and include exactly one stop transition for each controllable component plus a stop "global" transition. The generation of the MDP corresponding to a given MDPN has been described in [3]: it consists of (1) a composition step, merging the two subnets in a single net, (2) the generation of the reachability graph RG of the composed net, (3) two reduction steps transforming each PR and ND sequence in the RG into a single MDP transition.

In the next subsections a pattern based approach to generate a MDPN mimicking the dynamic behavior of an NdRFT is presented. We introduce the set of repairable basic components: $\underline{\mathcal{E}}_R = \{e \in \underline{\mathcal{E}} | e.rep = true\}$, the set of internal events with global repair strategy $\overline{\mathcal{E}}_{GR} = \{e \in \overline{\mathcal{E}} | e.obs = true \wedge e.str = global\}$ and the set $Comp^{pr} = \underline{\mathcal{E}} \cup \overline{\mathcal{E}}_{GR}$ of components of the MDPN and the subset $Comp^{nd} = \underline{\mathcal{E}}_R \cup \overline{\mathcal{E}}_{GR}$ of controllable components.

The PN^{pr} and the PN^{nd} are obtained directly from the NdRFT model using a pattern-based approach. We illustrate the method describing the basic patterns, and how to instantiate and compose them.

4.1 Generating the PR subnet

Fig. 2 shows how each BE can be translated in a PN^{pr} submodel according to their *rep* attribute: each non repairable event is translated into subnet **A** while each repairable event is translated into subnet **B**. It is easy to recognize the places that model the state of each (basic) event e labeled UP_e, $DOWN_e$ and $UnderRepair_e$ (actually when the $UnderRepair_e$ is marked, also the $DOWN_e$ place is marked, until the repair process ends). Run and Stop transitions have different icons, so that they can be easily distinguished. Moreover each transition has a priority (label $prio_i$ indicated next to each transition) and a weight, that is renormalized w.r.t. the set of enabled transitions to obtain a firing probability. At each probabilistic step an Up component can either remain Up (sequence $WorkR_e$, $WorkS_e$) or go $Down$ (sequence $FailR_e$, $FailS_e$); each transition participating to this first step involves only one component, namely e. The chosen priority assignment is due to the way basic event states are propagated to obtain intermediate event states, as will be discussed later. A $Down$ component can either remain $Down$ (stop transition $FailS_e$) or start its repair (run transition $Repair$, either followed by the sequence $ContRepR_e$ and $ContRepS_e$, meaning that the repair has not completed in the current time unit, or by the sequence $EndRepR_e$, $EndRepS_e$ if the repair completes). Place $Assign_e$ is set by the PN^{nd} when a decision to repair e is taken. Places AV_RES_i represent the resources, and they become available as the repair ends. A token in place $NotInvolved_e$ means that the component corresponding

to the BE is not involved in any repair action. The *rprob* and $fprob$ attributes associated with the events are used to properly weight the transitions representing failure and end/continuation of repair actions: $fprob$ is associated with transition $FailR_e$, $1 - fprob$ is associated with transition $WorkR_e$, *rprob* (representing the probability of continuing to repair) is associated with transition $ContRepR_e$, finally $1 - rprob$ is associated with transition $EndRepR_e$.

Observe that the only effective conflicts to be resolved on the PN^{pr} model are the following free choice conflicts: $WorkR_e$ vs. $FailR_e$ (for each basic event), $ContRepR_e$ vs. $EndRepR_e$ (for each locally repairable basic event), plus the free choice conflict $ContRepGR_e$ vs. $EndRepGR_e$ for each global repair action (whose translation pattern is commented hereafter). Hence the weight assigned to all other transitions are irrelevant, since they will eventually fire once enabled (i.e. their firing is not the result of a conflict resolution).

Let us now discuss the translation pattern ensuring the propagation of the state from basic to internal events, and of the global repair actions, associated with some internal event.

The conversion rule for an AND/OR gate corresponding to a given internal event e is shown in Fig. 3. Subnets **C** and **E** simply model the propagation of the faults from the input events of the gate to its output event. These patterns are actually "templates" that must be instantiated according to the set of inputs of the AND/OR gate, which in general includes a subset of internal events, and a subset of basic events. The components involved in the firing of transition AND_e are all basic events which are the leaves of the subtree originating in the AND gate. The components involved in the firing of each transition OR_i are: the basic event e_j if the input place of OR_i is $DOWN_{e_j}$, otherwise, if the input place of OR_i is place $OUTCOMP_{e_k}$, the involved components correspond to the set of basic events which are the leaves of the subtree originating in the internal event e_k. All these "state propagation" transitions are "run" and must fire after all decisions about occurrence of failures and end of repair have been taken for all basic events (and global repair internal events), but before the stop transitions have fired for any basic event (in fact all stop transitions have a lower priority level than the propagation transitions).

Internal events that are not observable or have local repair strategy are translated into these simple subnets.

Those with a global repair strategy have an additional subnet **D** (common to both gate types) shown on the right of Fig. 3: this subnet represents the corresponding global repair process. In particular, place $IdleSupervisor_e$ is marked when no global repair process involving the supervisor internal event e has started yet. The start of a global repair process, represented by the firing of the "run" transition $RepairG_e$, involving component e, is enabled when the $Assigned_e$ place is marked (indicating that the PN^{nd} subnet, in the previous ND step has decided to assign the required resources for such global repair process, to supervisor e). The firing of "stop" transition $EndG_e$, instead, means that no global repair supervised by e will start in the current time step: observe that this transition has lower priority than $RepairG_e$, hence the repair process

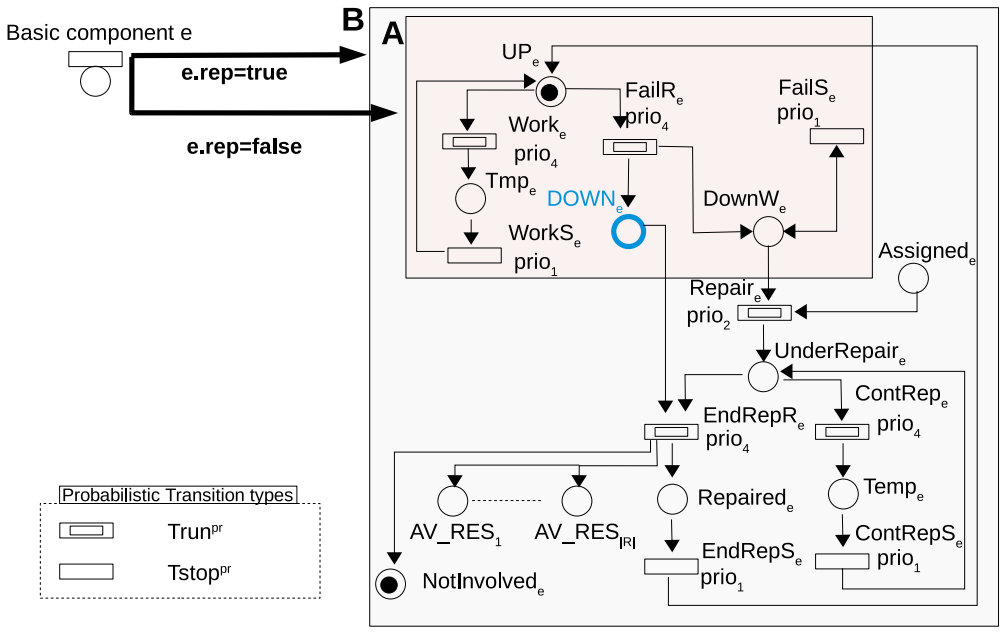

Figure 2: Conversion of the NdRFT BEs into submodels of PN^{pr} of an MDPN.

starts as soon as the required resources have been assigned to supervisor e. Place $UnderRepair_e$ represents the fact that the global repair process supervised by e is ongoing: if the "run" transition $ContRepGR_e$ fires, followed by the "stop" transition $ContRepGS_e$, the repair process will not end in the current time step, while if the "run" transition $EndRepGR_e$ fires (setting the resources free), followed by the "stop" transition $EndRepGS_e$, the global repair process will end in the current time step (all the above mentioned transitions, involve component e): this triggers the firing of the transitions $ResetR_{ei}$, $ResetS_{ei}$ or $FreeR_{ei}$, $FreeS_{ei}$, (all involving basic component e_i in the set of basic events supervised by e in the global repair process) ensuring that all basic events involved in the repair process are reset to the Up state (place UP_{ei} marked) and the corresponding $NotInvolved_{ei}$ place is marked again.

The translation algorithm visits all the events in the NdRFT and generates for each of them an appropriate PN submodel (the selection of the appropriate PN submodel follows the indications depicted in the template figures). Finally all submodels are composed by merging the places with equal label, leading to the whole probabilistic subnet of the MDPN.

Algorithm 1 shows how to generate the corresponding PN^{pr} from the RNdRFT. The set $PNet$ is used to store all the PN submodels generated by the algorithm during intermediate steps. In the end it will contain a single element: the PN^{pr}.

The algorithm visits all the events in the RNdRFT and inserts for each of them an appropriate PN submodel in $PNet$. The selection of the appropriate PN submodel corresponding to the event type is computed

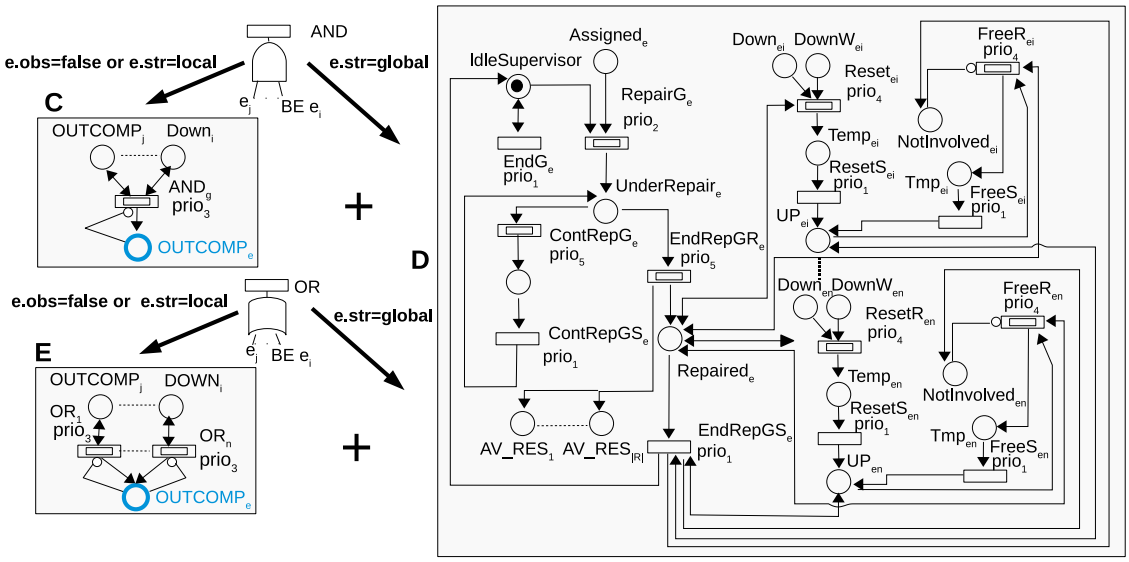

Figure 3: Conversion of the NdRFT AND/OR gate plus its output event into submodels of PN^{pr} of an MDPN.

by the function $PN(e, type)$.

Finally the method $Compose()$ substitutes all the submodel in $PNet$ are composed by merging the places with equal label, leading to the whole probabilistic subnet of the MDPN.

4.2 Generation of the ND subnet

The corresponding PN^{nd} is built from the template subnets depicted in Fig. 4 and 5. The basic idea is that the PN^{nd} submodel must decide whether a repair action must be started for each down BE and for each observable internal event which may trigger a global repair process. For any repairable BE e (corresponding to a controllable component in the MDPN), firing of stop transition $NoAssign_e$, involving only component e, means that a non repair decision has been taken for event e, while firing of stop transition $Assign_e$, also involving only component e, corresponds to the opposite decision: observe that the second decision can be taken only if e is observable and in state $Down$, the needed resources are available (input places AV_RES_i contain enough tokens) and the event is not involved in any global repair process (input place $NotInvolved_e$ marked). The start of local repair actions triggered by a down and observable BEs is modeled by subnet G, right part. The start of local repair actions triggered by observable internal events is modeled by subnet L, where it is possible to observe the repetition of subnet G for as many times as the number of local repairs potentially triggered by the internal event e (the test arcs from place $OUTCOMP_e$ to the $Assign_{ei}$ transitions model the fact that the repair can start only if the internal event e is $Down$). Finally the start of a global

Algorithm 1: Algorithm for PN^{pr} generation from NdRFT

```
Class PN^pr GeneratePN^pr(Class NdRFT S )
Input: S is a NdRFT model
Output: A PN^pr model
set PNet= ∅;
set Events= insert_events(S);
while Events ≠ ∅ do
    e =Events.pick();
    if (e ∈ E_R) then PNet.insert(PN(e, B));
    else
        if (e ∈ E − E_R) then PNet.insert(PN(e, A)) else
            g =• e;
            if (g.type = AND) then
                if (e.str = local) then PNet.insert(PN(e, C));
                else PNet.insert(PN(e, C + D));
            end
            else
                if (e.str = local) then PNet.insert(PN(e, E));
                else PNet.insert(PN(e, E + D));
            end
        end
    end
end
PNet.Compose();
return PNet.pick();
```

repair action triggered by an internal event e is modeled by subnet I: it is possible to observe that a global repair process requires a single set of resources, starts for the set of supervised BEs as a whole, and requires that none of the supervised BEs be involved in any other repair process; on the other hand a local repair action triggered by an internal event e may start in different time steps for each basic event supervised by e (as long as e is still down and the conditions to start the local repair are satisfied). The two stop transitions $Assign_e$ and $NoAssign_e$ represent the two possible choices: each of them involves only component e.

Subnet H (as well as the $RUNGL_e$ transition in subnet L) is needed for technical reasons: it is used to "clear" the state of the internal events which must be recomputed at the end of each probabilistic step (after all fail/repair choices have been taken for all BEs and the continue/end of repair choices have been taken for all ongoing global repairs).

Again the final PN^{nd} submodel is obtained by properly composing the subnets generated for each event in the NdRFT and the special transition $StopGL$ (subnet M) used to conclude the non deterministic phase of the global system. It is worth noting that during the composition phase the places and the transitions with the same name are merged.

Finally in order to analyze the MDPN model, one has to define its reward functions. They are defined as follows: $rs(TE) = -1$ otherwise 0; $\forall t \in T^{nd}, rt(t) = 0$; $r_g = sum(rs, rt)$.

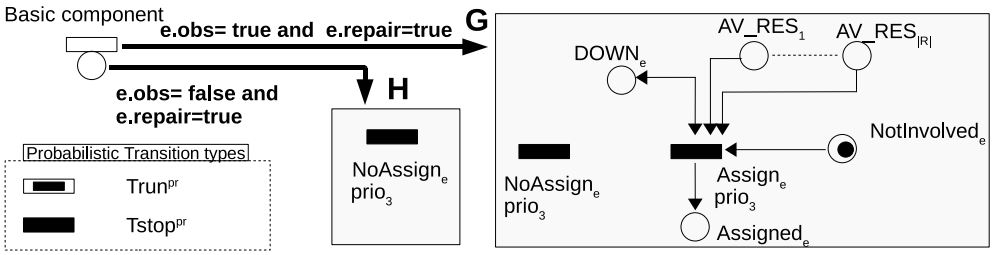

Figure 4: Conversion of the NdRFT BEs into submodels of PN^{nd} of an MDPN.

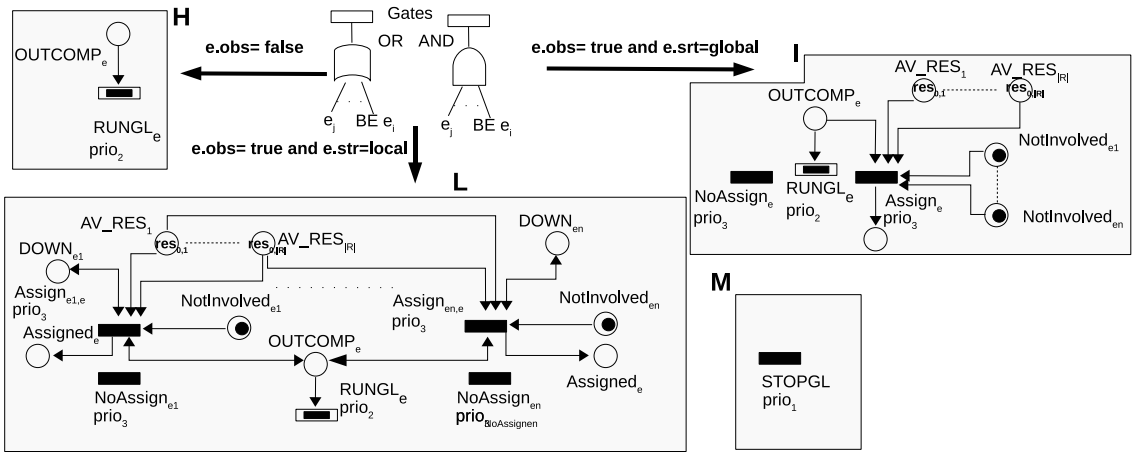

Figure 5: Conversion of the NdRFT gate into submodels of PN^{nd} of an MDPN.

This means that a negative reward (corresponding to a penalty) is associated with each state where the TE is *Down*. All other states and all actions have reward of 0. This means that every time unit spent in a state where TE is *Down* gives a penalty of -1. The optimization problem hence consists in finding the strategy that maximizes the reward (i.e. that minimizes the penalty).

More complex reward structures can be naturally devised to take into account the cost of repair actions, as well as the penalties due to the fact that the system is in a degraded state (the system is up but some subsystem is down, e.g. corresponding to a system with degraded performance).

Algorithm 2 shows how to generate the corresponding PN^{nd} from the RNdRFT..

In the same way, as in the algorithm 2, the set $PNet$ is used to store all the PN submodels generated by the algorithm during intermediate steps, and the function $PN(e, type)$ is used to select the appropriate PN submodel corresponding to the event type.

Algorithm 2: Algorithm for PN^{nd} generation from RNdRFT

```
Class PN^nd GeneratePN^nd(Class NdRFT S)
Input: S is a NdRFT model
Output: A PN^nd model
set PN=∅;
set Events=insert_events(S);
while Events ≠ ∅ do
    e =Events.pick();
    if (e ∈ E_R and e.obs) then PNet.insert(PN(e, G));
    else
        if e ∈ Ē then
            if (!e.obs) then PNet.insert(PN(e, H));
            else
                if (e.obs and e.str = global) then PNet.insert(PN(e, I));
                else PNet.insert(PN(e, L));
            end
        end
    end
end
PNet.insert(PN(NULL, STOPGL));
PNet.Compose();
return PNet.pick();
```

4.3 Translation correctness

Let us prove that the MDPN obtained by applying the above translation procedure produces a Reachability Graoh (RG) from which it is possible to derive the MDP corresponding to the NdRFT semantics, defined in Section 3.2.

To this purpose we must define *maximal* non deterministic or probabilistic firing sequences of a MDPN.

Definition 4 *A maximal non deterministic firing sequence (MNDFS) is characterized by the following properties: (1) it starts either in the initial state or in a state reached by a maximal probabilistic firing sequence, (2) it contains exactly one stop transition for each controllable component, and one "global" stop transition.*

Definition 5 *A maximal probabilistic firing (MPRFS) sequence is characterized by the following properties: (1) it starts in a state reached by a maximal non deterministic firing sequence, (2) it contains exactly one stop transition for each component.*

In the sequel the correspondence between MDPN and MDP states as well as the correspondence between MNDFS and MPRFS in the MDPN and MDP actions and probabilistic transitions are stated and proven.

MDPN states vs. MDP states First of all we need to define the correspondence between a subset of states appearing in the RG of the MDPN (both those reached immediately after the firing of a MPRFS and those reached immediately after a MNDFS, i.e. an action) and the MDP states.

The state of each BE e (Up, $Down$, $LocRep$, $GlobRep_*$) is represented by the following places:

- $st_e = Up$ if place UP_e is marked;
- $st_e = Down$ if place $DOWN_e$ is marked and place $NotInvolved_e$ is marked;
- $st_e = LocRep$ if place $UnderRepair_e$ is marked;
- $st_e = GlobRep_d$ ($st_e = GlobRep_u$) when place $DOWN_e$ (UP_e) is marked, places$UnderRepair_e$ and $NotInvolved_e$ are not marked; in this case there must exist exactly one internal event e' s.t. $e'.obs = trueande'.str = globalande \in e'.torep$ and place $UnderRepair_{e'}$ is marked, so that $sup_e = e'$.

The $Up/Down$ state of internal events are derived according to the FT structure (represented by subnets C and E in Fig.3): an IE e is Down if place $OUTCOMP_e$ is marked at the end of a MPRFS.

It is thus possible to establish a correspondence between each non deterministic marking m (reached immediately after the firing of a maximal probabilistic transition sequence) of the MDPN and a state ρ of the MDP: we use the notation m_ρ to indicate a non deterministic marking of the MDPN corresponding to state ρ of the MDP. Similarly it is possible to establish a correspondence between each intermediate state $\langle \rho, a \rangle$ of the MDP and a marking m' reached immediately after the firing of a maximal non deterministic transition sequence of the MDPN. The set of resources in use, expressed by res_ρ in the MDP, is represented in the MDPN by resource-indexed places AV_RES_r: the initial marking of AV_RES_r corresponds to the multiplicity of resource r in res_0, while the set res_ρ of resources in use in state ρ corresponds to: $res_\rho(r) = res_0(r) - m_\rho(AV\_RES_r)$.

The initial marking, corresponding to the initial MDP state, has one token in each place UP_e and as many tokens as the number of available resources of type r in places AV_RES_r.

In order to prove that the translation is correct, we have to show that there is a one to one correspondence between the actions $a \in A_\rho$ and the MNDFS σ_a enabled in m_ρ, that the intermediate state $\langle \rho, a \rangle$ corresponds to the marking $m_{\langle \rho,a \rangle}$ reached by firing σ_a in m_ρ. Moreover there is a correspondence between the states reachable from the intermediate state $\langle \rho, a \rangle$ and those reachable from $m_{\langle \rho,a \rangle}$ through some MPRFS, finally the probability of transition $\langle \rho, a \rangle \to \rho'$ is equal to the sum of probabilities associated with the set of MPRFS leading from $m_{\langle \rho,a \rangle}$ to $m_{\rho'}$.

MDPN non deterministic sequences vs. MDP actions From a non deterministic state of the MDPN, one or more alternative MNDFS may fire, each comprising exactly one stop transition for each controllable component (BE repairable event or IE event with global repair strategy) plus one global stop transition: the combination of all stop transitions in each MNDFS defines a possible action at the MDP level, corresponding

to the set of decisions - start repair of component e ("stop" transition $Assign_e$) or do not start repair of component e ("stop" transition $NoAssign_e$) - for each controllable component.

The set of decisions characterizing a specific action a causes a state change (corresponding to the Non Deterministic step described in Section 3.2) witnessed by the marking of the $Assigned_e$ places in the MDPN at the end of the corresponding MNDFS σ_a. It is easy to see that the conditions expressed in Section 3.2 for moving a BE e from the *Down* state to the *LocRep* state or for moving a set of BE to their current state to the $GlobRep_*$ state (provided they are in the *torep* set of a *Down* IE e') correspond to the conditions for firing transitions $Assign_e$ or $Assign_{e'}$ in the PN^{nd} subnet.

In fact, if we consider subnet G in Figure 4, corresponding to an observable and repairable BE e, a decision to start (local) repair may be taken if (1) e is in state *Down*, (2) the required resources are available, and (3) the component is not yet involved in any other repair action. As an alternative, if the BE e is repairable but not observable, and it is in the *torep* set of some internal event e' with local repair strategy, then the local repair can start if (1) both the BE e and the internal event e' are *Down*, (2) the required resources are available, and (3) e is not yet involved in any other repair action: this is modeled by subnet L in Figure 5.

The state change from state *Down* to state $GlobRep_*$ instead is modeled by subnet I in Figure 5, and corresponds to the firing of the stop transition $Assign_e$ where e is an observable internal event with associated global repair strategy: this transition may occur only when IE e is in state *Down*, the resources needed for the global repair are all available, and none of the BE in *e.torep* are involved in any other repair process (places $NotInvolved_{ei}$ marked). Observe that the start of global repair for internal event e actually causes all the BEs in *e.torep* to switch to the $GlobRep_*$ state simultaneously.

Since a decision is necessarily taken for any controllable component (exactly one stop transition must fire for each controllable component in any MNDFS), and since for each controllable event is always possible to take a NoAssign decision, and if the state allows so it is also possible to take the alternative Assign decision, then all possible combination of start/do not start repair decisions corresponding to the allowed actions in the MDP can be obtained, and due to the conditions on the Assign transitions no combination of decisions corresponding to an impossible action can be fired in the MDPN.

MDPN probabilistic sequences vs. MDP probabilistic state change following an action After each MDP action a probabilistic state change occurs: in the MDPN this corresponds to the MPRFS that may follow a MNDFS. The probability of each path is obtained as the product of the probability associated with each transition in the path. Observe that a probabilistic path can be described as the interleaving of several subpaths, one for each component e represented in the MDPN, and ending with a stop transition involving e. The transitions firing in each subpath depend on the initial status of the component:

- if e is Up (place UP_e marked) and not involved in any global repair action (place $NotInvolved_e$ marked)

then either the subpath contains the sequence $WorkR_e$, $WorkS_e$, or it contains the sequence $FailR_e$, $FailS_e$: the former doesn't cause a state change for e, while the second corresponds to a change from Up to $Down$. Observe that if e is Up but it is involved in a global repair process, then the subpath for e depends on the probabilistic evolution of its supervisor, hence this case will be discussed together with such evolution;

- if e is $Down$ (place $DOWN_e$ marked) and not involved in any repair action (place $NotInvolved_e$ marked), the subpath shall include only the stop transition $FailS_e$, which does not cause any state change (e remains $Down$);

- if e is in state $LocRep$, which includes also the case of place $Assigned_e$ just marked by the last non deterministic sequence (hence allowing run transition $Repair_e$ to fire thus marking the $UnderRepair_e$ place) then either the repair process continues (sequence $ContRepR_e$,$ContRepS_e$), thus leaving the component in the $LocRep$ state, or the repair process ends (sequence $EndRepR_e$,$EndRepS_e$), which causes e to come back to the Up state;

- global repair processes influence the state of the supervised basic events; if a given internal event e' with global repair strategy is $Down$, and the basic events in $e'.torep$ are non involved in any repair process, it can be assigned the resources for the corresponding global repair to start (place $Assigned_{e'}$ marked at the end of a MNDFS): as a consequence the global repair process can start (transition $RepairG_{e'}$, marking the place $UnderRepair_{e'}$), causing a state change of the BEs in $e'.torep$ to $GlobRep_*$ (where $*$ stands for u or d depending on the previous BE status). As in the case of local repair, the $GlobRep$ state is an intermediate one, which can become stable if the repair does not end in the current time step (transitions $ContRepGR_{e'}$, $ContRepGS_{e'}$), while in case the repair process ends (transitions $EndRepGR_{e'}$, $EndRepGS_{e'}$) then all the BEs supervised by e' are reset to the Up state: this is achieved by firing the transitions $ResetR_{ei}$, $ResetS_{ei}$, or transitions $FreeR_{ei}$, $FreeS_{ei}$ (the last two transitions fire in case UP_{e_i} is already marked, to reset the marking of $NotInvolved_{ei}$). Observe that in any case all BEs supervised by e' switch from to $GlobRep_*$ simultaneously, and from $GlobRep_*$ to Up (or from $Down$ to Up, if the repair process lasts only one time step) simultaneously.

  It may be the case that while a global repair process is ongoing, some of the BEs in $e'.torep$ that were not faulty, fail in the current time step (transitions $FailR_{e_i}$, $FailS_{e_i}$ may fire if the choice of continuing the repair supervised by e' has already been taken, i.e. if transitions $ContRepGR_{e'}$, which has priority $prio_5$ has already fired). In this case their state changes from Up to $GlobRep_d$ (with a not observable passage through the $Down$ state), and will be reset to the Up state as soon as the global repair process ends (which might happen in the same time unit).

Observe that the possible state changes described above for each component, are exactly the same illustrated in the probabilistic step of the MDP semantics of the NdRFT (see Section 3.2).

The probability of each possible MPRFS is obtained as a product of the normalized weight of the enabled transitions. Observe that transitions corresponding to different components are never in conflict (the failure or end-repair choice of one component cannot influence the choice of any other component, by construction): this means that independently on the chosen interleaving order of the components, the overall probability moving from a given marking m to a new marking m' only depends on the failure probability of the Up components, and end-repair probability of the components under repair. If the priorities are set so that a specific order is forced in the MDPN there will be a single MPRFS leading from a given state m (corresponding to a MDP intermediate state $\langle \rho, a \rangle$) to a new marking m' (corresponding to a MDP state ρ'), and its probability will be exactly the same as that of the probabilistic step from $\langle \rho, a \rangle$ to ρ'. If instead priorities are set so that alternative interleavings may be chosen, leading from m to m', the final result will not change: indeed it is easy to show that the sum of the probabilities of the set of MPRFS leading from m to m' can be expressed as the product of the probabilities of the choices taken in each component (which are necessarily the same since the initial and final markings are the same) multiplied by a summation of probabilitites that sum up to one (these are the relative weights of the possible interleaving, which eventually converge to the same final state).

Finally observe that each MPRFS comprises a subsequence of AND_e and OR_e transitions, which are needed to propagate the correct Up or $Down$ state (place $OUTCOMP_e$ unmarked or marked respectively) of all IEs. This subsequence is deterministic (although different interleavings could be possible depending on the priority assignment) since it depends only on the state of the BEs, and hence it contributes as a factor 1 to the probability product. Observe that the priorities of these transitions are set so that they are fired after all probabilistic choices have been made, but before the firing of the stop transitions of all components.

This completes the proof. In fact, from the initial marking the set of possible actions in the MDP are in one to one correspondence with the MNDFS of the MDPN, the reached intermediate states are in one to one correspondence, and from these the same probabilistic state changes may occur, leading to corresponding new states. This also indicates how the MDP can be derived from the MDPN RG, only the markings from which maximal firing sequences are originated are kept, and the maximal firing sequences are substituted with the corresponding transitions in the MDP.

4.4 An transition priority assignment improving efficiency

In this subsection we will propose a method to associate priorities with the MDPN transitions, in order to reduce the possible interleavings corresponding to the same MPRFS or MNDFS: this improves the efficiency

of the method since it produces a reduction of the number of states of RG[3].

The method requires to fix a strict total order on the NdRFT events that must be compatible with the partial order induced by the NdRFT structure that is based on the dependencies: the TE is the lowest among all the events and two events are in relation $e < e'$ if e' is in the subtree of e. For any basic event e it cannot be in relation $e < e'$ w.r.t. any other internal event e'. The total order can be specified through an injective function ($ord : \mathcal{E} \rightarrow \mathbb{N}$) so that $\forall e, e' e < e' \Rightarrow ord(e) < ord(e')$.

Hence the priority assignment for the PR subnet is defined as follows:

- $\forall t_e \in Tstop^{pr} \Rightarrow prio_{t_e} = ord(e)$, where e is an event in $Comp^{pr}$
- $\forall t_e \in Trun^{pr} \Rightarrow prio_{t_e} = ord(e) + |Comp^{pr}|$, where e is an NdRFT event.

Observe that since the basic events have surely an higher $ord(e)$ value with respect to all internal events, the transitions that are used to update the state of the internal events will fire after all fail/do not fail and conclude repair/continue repair decisions have been taken for all basic events.

A similar priority assignment method is adopted for the ND subnet.

- $\forall t_e \in Tstop^{nd} \Rightarrow prio_{t_e} = ord(e)$, where e is an event in $Comp^{nd}$;
- $STOPGL \Rightarrow prio_{STOPGL} = Max_{e \in \mathcal{E}}(ord(e)) + 1$;
- $\forall RunGL_e, RunGL_{e'} \in Trun^{nd} \Rightarrow prio_{RunGL_e}, prio_{RunGL_{e'}} > Max_{e \in \mathcal{E}}(ord(e)) + 1 \wedge prio_{RunGL_e} \neq prio_{RunGL_{e'}}$.

The experiments presented in Sec. 6 have been performed applying the priority assignment method described above.

5 Framework architecture

The architecture of our framework for the NdRFT design and solution, is depicted in Fig. 6 and extends the one presented in [2], by introducing the new module *NdRFT2MDPN* able to convert an NdRFT model into MPDN, according to the conversion rules defined in Sec. 4. The solution process of an NdRFT model comprises five steps:

1. The NdRFT model drawn by the user by means of *Draw-Net* [11], is stored in a XML file (.mdl) and becomes the input of *NdRFT2MDPN*. The resulting MDPN model consists of two separate *Petri Nets* (PN):

[3]Observe that the assignment of a different priority level to a pair of transitions that cannot be in conflict is irrelevant; on the other hand if the transitions are potentially in conflict then their priority assignment can constrain the set of possible strategies that will be considered at the MDP level and may exclude the optimal one.

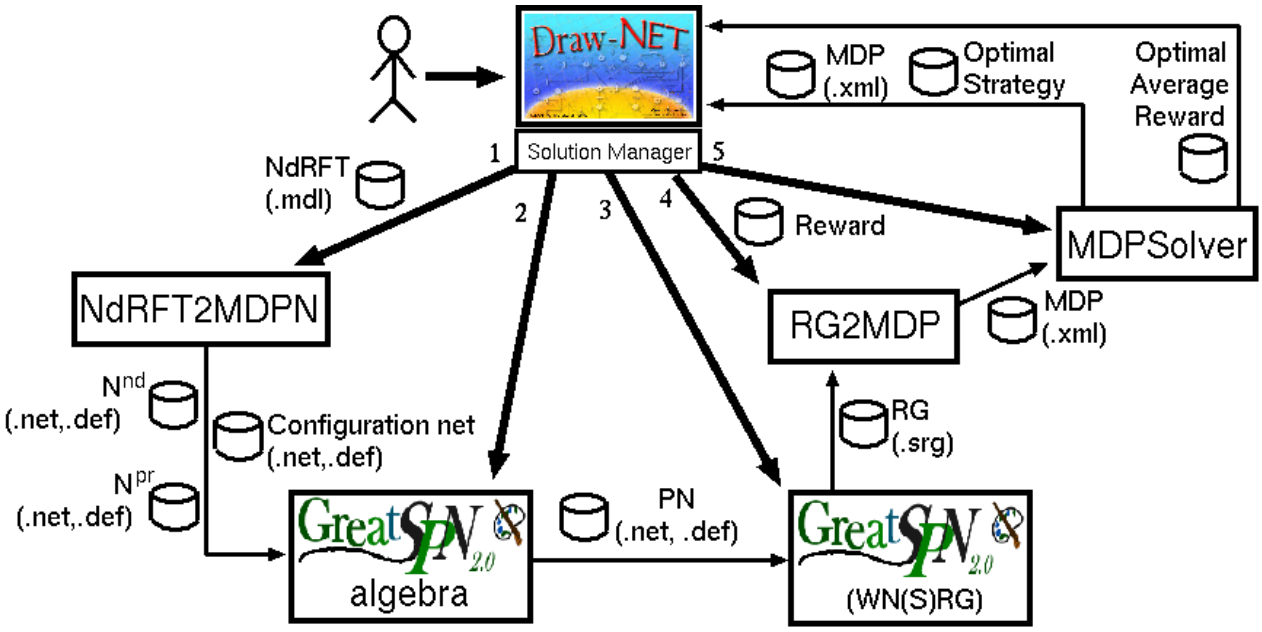

Figure 6: Framework architecture.

the probabilistic PN (N^{pr}) and the non deterministic PN (N^{nd}); each of these nets is stored in a couple of files (.net, .def) according to the *GreatSPN* [10] file format.

2. The N^{pr} and the N^{nd} models are composed by place merging; this is done by means of the *algebra* tool [10]. The result of this step is a Petri Net (PN).

3. The PN is the input of *WN(S)RG* generating the *Reachability Graph* (RG) [9]. The resulting graph is stored in a specific file (.srg).

4. From the graph obtained in step 3, an MDP is derived by means of the *RG2MDP* converter.

5. The obtained MDP is stored in an XML file which is in turn processed by the *MDPSolver* producing the *optimal repair strategy*. According to such strategy the system unavailability can be computed. Both results can be visualized by *Draw-Net*.

6 Experiment results

The example we report is inspired to the *Active Heat Rejection System* (AHRS) presented in [1]. The block scheme of our version of the AHRS's architecture is depicted in Fig. 1.a: the system is composed by three redundant thermal rejection units $U1$, $U2$ and $U3$. $U1$ is composed by the heat source $A1$ and the power source $P1$. Similarly, $U2$ is composed by $A2$ and $P2$, while $U3$ by $A3$ and $P3$.

Fig. 1.b shows the NdRFT model for the AHRS system; the failure probability ($\downarrow$) and the repair probability ($\uparrow$) of each basic component are shown in the same figure. The unit $U1$ fails if its heat source $A1$ is failed or if its power source $P1$ is failed. Similarly, the failure of $U2$ and $U3$ is due to the failure of their respective heat source or power source. The failure of the whole system (TE) occurs if all the thermal rejection units are failed.

The NdRFT model in Fig. 1.b shows that in our version of the AHRS, several components are repairable ($A1$, $P1$, $A2$, $P2$), whereas their failure can be observable or not. Two repair processes can be activated: 1) a global repair process in case of failure of $U2$ and involving the components $A2$ and $P2$; 2) a local repair process in case of the system failure (TE) and involving the components $A1$, $P1$, $A2$ and $P2$. In case of global repair, one repair resource is used to repair the subsystem; in case of local repair instead, one resource has to be dedicated to the repair of each component of the system. We suppose that in our case study, two repair resources are available (Fig. 1). One resource is used for the global repair of $U2$: while such repair process is running, the local repair of the system (TE) may start but in this case, it can exploit only one resource because the other one is already is used in the global repair of $U2$. So only one component ($A1$ or $P1$) could be locally repaired during the global repair of $U2$. If instead the local repair of the system starts while the global repair of $U2$ is not running, then the local repair can exploit both resources and two components among $A1$, $P1$, $A2$, $P2$ can be repaired at the same time. In this case, during the local repair of the system, the global repair of $U2$ can not run since all the resources are already in use.

The RG of the MDPN model obtained by the NdRFT in Fig. 1.b has 11.515 states; while the underlying MDP has 389 states. This difference in terms of number of states between the RG of the MDPN[4] and the obtained MDP is due to the fact that the MDPN formalism gives a macroscopic view of probabilistic and non deterministic behaviors of the system. In other words, at MDPN level, complex non deterministic and probabilistic behaviors are expressed as a composition of simpler non deterministic or probabilistic steps, that will be reduced to a single step in the final MDP.

Since the non repairable components ($A3$ and $P3$) cannot induce directly the failure of the global system, we can compute the average reward and the optimal strategy of the underlying MDP at infinite horizon. Observe that defining the optimal strategy for this model is not trivial: for instance when all the basic events are down then the optimal strategy suggests us to repair $P1$ with a local repair action, while $A2, P2$ with a global repair action. This is justified by the fact that the global repair action of $A2, P2$ needed only one resource. Instead when $A1, P1, P2$ and $P3$ are down, it suggests to repair $P1$ and $P2$ with a local repair action. The choice to repair locally $P2$ is justified by the fact that in this case the probability to repair the component ($1 - P2.rprob$) in one time unit is greater than that associated with the global repair action ($1 - U2.rprob$).

[4]We recall that the RG of the MDPN model is used in the reduction step for obtaining the MDP as described in [3]

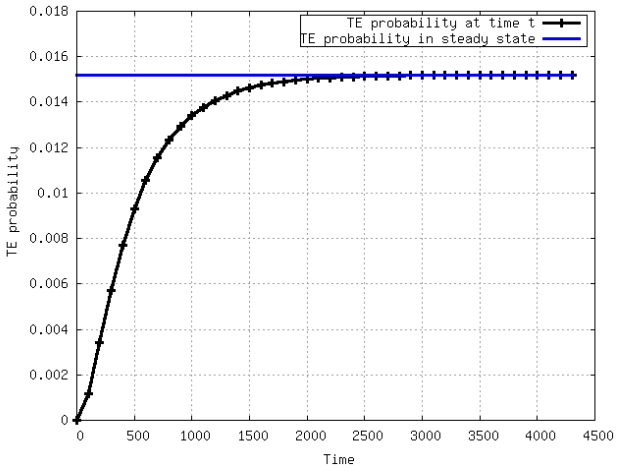

Figure 7: TE probability at time t increasing the t values ($0 \leq t \leq 4500$)

Table 1: Experiments increasing the example size

	RG		RRG	MDP	
Com.	St.	Time	St.	St.	Time
1,1,1	11.515	1s	5.262	389	0
2,1,1	50.844	7s	21.094	937	6s
2,2,1	921.354	167s	401.350	7.754.	1.630s
2,2,2	16.841.490	≈23h	6.048.310	32.558	≈4h
Using priorities among the transitions					
1,1,1	3.189	0	1.572	389	0
2,1,1	35.555	8s	11.581	937	4s
2,2,1	453.257	145s	147.716	7.754	1.614s
2,2,2	2.919.999	≈2h	1.048.310	32.558	7.006s

Moreover we have computed the TE probability in steady state, solving the DTMC obtained from the underlying MDP fixing the action to take in every state according to the performed optimal strategy. In particular we have obtained that the TE probability of this model in steady state is 0.0151943. We have also studied the TE probability at time t, so that we have observed that this probability converges to the steady state probability. TE probability at time 4300 is equal to TE probability in steady state as shown in Fig. 7.

Finally the Tab. 1 shows some experiments performed increasing the dimension of our example. Practically we have replicated the subtrees of the NdRFT model in Fig. 1.b. For instance, in Tab. 1, $2, 2, 2$ means that we have duplicated the subtrees rooted in $U1$, $U2$, $U3$ respectively, while $1, 1, 2$ means that we have duplicated only the subtree of $U3$.

The computation has been performed with an INTEL Centrino DUO 2.7 of 2Gb memory capacity. In

particular the first column shows the model complexity, the second and the third one the RG number of states and its computation time, the fourth the RRG number of states, and the last two columns the MDP number of states and its generation and solution time.

These results show that state space grows very fast (the state space explosion problem), so that the model becomes quickly intractable. A further reduction of the number of states for this model can be achieved associating different priorities with the system transitions such that the number of possible interleavings of the non deterministic/probabilistic actions in each path are reduced (see the results in second part of Tab. 1). It is important to observe that a different priority level can be set up only among independent actions; in fact if the actions are not independent then all the priority can constrain the set of policies to be considered (and may exclude the optimal one).

Another possible way to mitigate the state space explosion problem consists in translating the NdRFT model into a *Markov Decision Well-formed Net model* (MDWN) [3]. From a MDWN, a reduced MDP can be obtained, and provides the optimal strategy equivalent to the one given by the not reduced MDP.

7 Conclusion and future work

We have defined a new FT extension called NdRFT that allows to model failure modes of complex systems as well as their repair processes. The originality of this formalism with respect to other proposals is that it allows to manage repair strategies optimization problems. This is done by defining the NdRFT semantics in terms of an MDP and then solving the optimization problem using the techniques available for MDPs. The generation of the MDP is achieved by an intermediate translation of the NdRFT model into an MDPN, so that we can reuse the efficient algorithms devised to derive an MDP from an MDPN. We have also highlighted that NdRFT allows to express in an elegant way several possible repair start options based on the following concepts: observability of events, the notion of local versus global repair action, the notion of repair supervisor component in case of global repair.

A possible future work is extending the NdRFT, so that the modeler can directly define more complex reward functions, for instance considering the cost of repair actions, or the penalties due to the fact that the system is in a degraded state (the system is up, but some subsystem is down, e.g. corresponding to a system with degraded performance).

Another future work in order to mitigate the well-known state space explosion problem in the final MDP, consists in translating the NdRFT model into a MDWN instead of an MDPN. In fact for MDWN, an efficient analysis technique taking advantage from the intrinsic symmetries of the system, is developed: from an MDWN it is possible to derive a *Symbolic Reachability Graph* [9], and from it a reduced MDP can be obtained. This allows a computational cost reduction, but the optimal strategy computed on the reduced

MDP is equivalent to the one computed on the ordinary MDP. This possibility is useful when the system is characterized by symmetries and redundancies in its structure.

Observe that the NdRFT formalism could be extended by considering *dynamic gates* [16], which allow to express functional and temporal dependencies among component failures, as well as repair resources preemption.

Acknowledgments The activity of M. Beccuti, G. Franceschinis and D. Codetta-Raiteri has been partially supported by the EU-Project CRUTIAL IST-2004-27513.

The activity of S. Haddad has been partially supported by Galileo project n. 17599NB and ANR project Checkbound ANR-06-SETI-002.

A Markov Decision Petri Net

A Markov Decision Petri Net $\mathcal{MN}$ is composed by two different parts (i.e. two extended Petri nets): the probabilistic one N^{pr} and the non deterministic one N^{nd} called the *decision maker*; it is thus possible to clearly distinguish and design the probabilistic behavior of the system and the non deterministic one. The probabilistic part models the probabilistic behavior of the system and can be seen as composition of a set of n components ($Comp^{pr}$) that can interact; instead the non deterministic part models the non deterministic behavior of the system where the decisions must be taken (we shall call this part *the decision maker*). Hence the global system behavior can be described as an alternating sequence of probabilistic and non deterministic phases.

The probabilistic behavior of a component is characterized by two different types of transitions $Trun^{pr}$ and $Tstop^{pr}$. The $Trun^{pr}$ transitions represent intermediate steps in a probabilistic behavior phase and can involve several components (synchronized through that transition), while the $Tstop^{pr}$ ones always represent the final step of the probabilistic phase of at least one component.

In the non deterministic part, the decisions can be defined at the system level (transitions of T_g^{nd}) or at the component level (transitions of T_l^{nd}). The sets T_g^{nd} and T_l^{nd} are again partitioned in $Trun_g^{nd}$ and $Tstop_g^{nd}$, and $Trun_l^{nd}$ and $Tstop_l^{nd}$ with the same meaning. The decision maker does not necessarily control every component and may not take global decisions. Thus the set of controllable "components" $Comp^{nd}$ is a subset of $Comp^{pr} \uplus \{id_s\}$ where id_s denotes the whole system.

The probabilistic net is enlarged with a mapping *weight* associating a weight with every transition in order to compute the probabilistic choice between transitions enabled in a marking. Furthermore it includes a mapping *act* which associates to every transition the subset of components that (synchronously) trigger the transition. The non deterministic net is enlarged with a mapping *obj* which associates with every transition

the component which is involved by the transition. The following definition summarizes and formalizes this presentation.

Definition 6 (Markov Decision Petri Net (MDPN)) *A Markov Decision Petri Net (MDPN) is a tuple* $\mathcal{MN} = \langle Comp^{pr}, Comp^{nd}, N^{pr}, N^{nd} \rangle$ *where:*

- $Comp^{pr}$ *is a finite non empty set of components;*
- $Comp^{nd} \subseteq Comp^{pr} \uplus \{id_s\}$ *is the non empty set of controllable components;*
- N^{pr} *is defined by a PN with priorities [17]* $\langle P, T^{pr}, I^{pr}, O^{pr}, H^{pr}, prio^{pr}, m_0 \rangle$*, a mapping weight:* $T^{pr} \rightarrow \mathbb{R}$ *and a mapping act:* $T^{pr} \rightarrow 2^{Comp^{pr}}$*. Moreover* $T^{pr} = Trun^{pr} \uplus Tstop^{pr}$
- N^{nd} *is defined by a PN with priorities* $\langle P, T^{nd}, I^{nd}, O^{nd}, H^{nd}, prio^{nd}, m_0 \rangle$ *and a mapping obj:* $T^{nd} \rightarrow Comp^{nd}$*. Moreover* $T^{nd} = Trun^{nd} \uplus Tstop^{nd}$*.*

Furthermore, the following constraints must be fulfilled:

- $T^{pr} \cap T^{nd} = \emptyset$*. A transition cannot be non deterministic and probabilistic.*
- $\forall id \in Comp^{pr}, \exists C \subseteq Comp^{pr}$*, s.t.* $id \in C$ *and* $act^{-1}(\{C\}) \cap Tstop^{pr} \neq \emptyset$*. Every component must trigger at least one* final *probabilistic transition.*
- $\forall id \in Comp^{nd}, obj^{-1}(\{id\}) \cap Tstop^{nd} \neq \emptyset$*. Every controllable component must be the object of at least one* final *non deterministic transition.*

Note that the probabilistic part and the decision maker share the same set of places and the same initial marking. Let us now introduce the rewards associated with the MDPN net. As will be developed later, an action of the decision maker corresponds to a sequence of transition firings starting from some marking. We choose to specify a reward by first associating with every marking m a reward $rs(m)$, with every transition t a reward $rt(t)$ and then by combining them with an additional function rg (whose first parameter is a state reward and the second one is a reward associated with a sequence of transition firings). The requirement on its behavior given in the next definition will be explained when presenting the semantics of a MDPN.

Definition 7 (MDPN reward functions) *Let* $\mathcal{MN}$ *be a MDPN. Then its reward specification is given by:*

- $rs : \mathbb{N}^P \rightarrow \mathbb{R}$ *which defines for every marking its reward value.*
- $rt : T^{nd} \rightarrow \mathbb{R}$ *which defines for every transition its reward value.*
- $rg : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$*, not decreasing w.r.t its second parameter.*

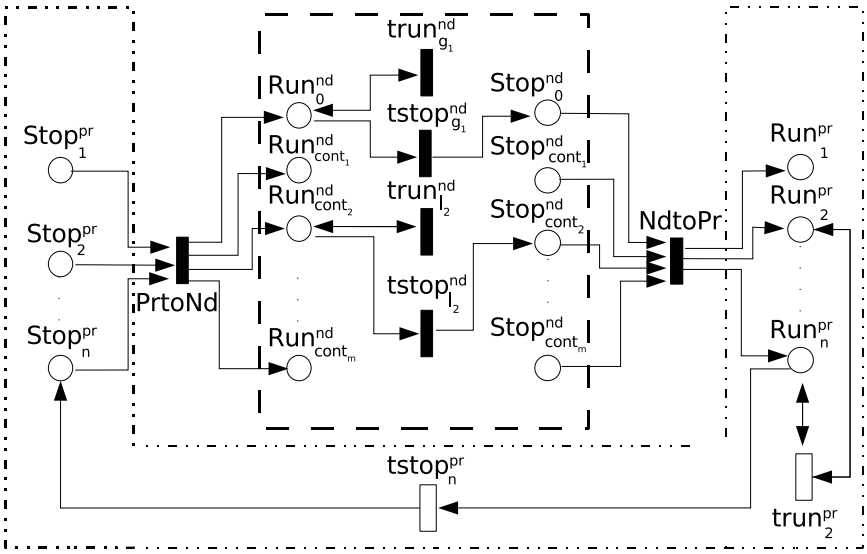

Figure 8: Arcs connecting the places $Stop_i^{pr}$, Run_i^{nd} and the transition $PrtoNd$; arcs connecting the places $Stop_i^{nd}$, Run_i^{nd} and the transition $NdtoPr$

A.1 MDPN semantics.

The MDPN semantics is given in three steps. First, one composes the probabilistic part and the decision maker in order to derive a unique PN. Then one generates the (finite) reachability graph (RG) of the PN. At last, one produces an MDP from it.

From MDPN to PN. First we explain the semantics of additional places $Stop_i^{pr}$, Run_i^{pr}, $Stop_i^{nd}$, Run_i^{nd}, $Stop_0^{nd}$ and $Run_0^{nd}$and additional non deterministic transitions $PrtoNd$ and $NdtoPr$. Places $Stop_i^{pr}$, Run_i^{pr}, $Stop_i^{nd}$, Run_i^{nd}, $Stop_0^{nd}$ and Run_0^{nd} regulate the interaction among the components, the global system and the decision maker. There are places Run_i^{pr}, $Stop_i^{pr}$ for every component i, while we insert the places Run_0^{nd} and $Stop_0^{nd}$ if the decision maker takes same global decision and the pair of places Run_i^{nd} and $Stop_i^{nd}$ for every controllable component $i \in Comp^{nd}$. Non deterministic transitions $PrtoNd$ and $NdtoPr$ ensure that the decision maker takes a decision for every component in every time unit: the former triggers a non deterministic phase when all the components have finished their probabilistic phase whereas the latter triggers a probabilistic phase when the decision maker has taken final decisions for every controllable component.

The scheme describing how these additional items are connected together and with the nets of the MDPN is shown in Fig. 8. The whole PN $N^{comp} = \langle P^{comp}, T^{comp}, I^{comp}, O^{comp}, H^{comp}, prio^{comp}, m_0^{comp} \rangle$ related to a MDPN $\mathcal{MN}$ is defined below.

- $P^{comp} = P \uplus_{i \in Comp_{pr}} \{Run_i^{pr}, Stop_i^{pr}\} \uplus_{i \in Comp_{nd}} \{Run_i^{nd}, Stop_i^{nd}\}$

- $T^{comp} = T^{pr} \uplus T^{nd} \uplus \{PrtoNd, NdtoPr\}$
- The incidence matrices of N^{comp} are defined by:
  - $\forall p \in P, t \in T^{nd}$,
    $I^{comp}(p,t) = I^{nd}(p,t), O^{comp}(p,t) = O^{nd}(p,t), H^{comp}(p,t) = H^{nd}(p,t)$
  - $\forall p \in P, t \in T^{pr}$,
    $I^{comp}(p,t) = I^{pr}(p,t), O^{comp}(p,t) = O^{pr}(p,t), H^{comp}(p,t) = H^{pr}(p,t)$
  - $\forall t \in Tstop^{pr}$ $s.t.$ $i \in act(t) : I^{comp}(Run_i^{pr}, t) = O^{comp}(Stop_i^{pr}, t) = 1$
  - $\forall t \in Trun^{pr}$ $s.t.$ $i \in act(t) : I^{comp}(Run_i^{pr}, t) = O^{comp}(Run_i^{pr}, t) = 1$
  - $\forall t \in Tstop^{nd}$ $s.t.$ $i \in act(t) : I^{comp}(Run_i^{nd}, t) = O^{comp}(Stop_i^{nd}, t) = 1$
  - $\forall t \in Trun^{nd}$ $s.t.$ $i \in act(t) : I^{comp}(Run_i^{nd}, t) = O^{comp}(Run_i^{nd}, t) = 1$
  - $\forall i \in Comp_{pr} : I^{comp}(Stop_i^{pr}, PrtoNd) = O^{comp}(Run_i^{pr}, NdtoPr) = 1$
  - $\forall i \in Comp_{nd} : I^{comp}(Stop_i^{nd}, NdtoPr,) = O^{comp}(Run_i^{nd}, PrtoNd) = 1$
  - for all $I(p,t), O(p,t), H(p,t)$ not previously defined,
    $I^{comp}(p,t) = O^{comp}(p,t) = 0, H^{comp}(p,t) = \infty$;
- $\forall t \in T^{nd}, prio(t) = prio^{nd}(t)$, $\forall t \in T^{pr}, prio(t) = prio^{pr}(t)$,
  $prio(PrtoNd) = prio(NdtoPr) = 1$, (actually these values are irrelevant)
- $\forall p \in P, m_0^{Comp}(p) = m_0(p)$, $m_0^{Comp}(Run_i^{nd}) = 1$,
  $m_0^{Comp}(Stop_i^{nd}) = m_0^{Comp}(Run_i^{pr}) = m_0^{Comp}(Stop_i^{pr}) = 0$.

RG semantics and transitions sequence reward. Considering the RG obtained from the PN we observe that the reachability set (RS) can be partitioned into two subsets: the non deterministic states (RS_{nd}), in which only non deterministic transitions are enabled, and the probabilistic states (RS_{pr}), in which only probabilistic transitions are enabled. By construction, the PN obtained from a MDPN can never reach a state enabling both nondeterministic and probabilistic transitions. A probabilistic transition can be enabled only if there is at least one place Run_i^{pr} with $m(Run_i^{pr}) > 0$, while a non deterministic transition can be enabled only if there is at least one place Run_i^{nd} with $m(Run_i^{nd}) > 0$. Initially only Run_i^{nd} places are marked. Then only when all the tokens in the Run_i^{nd} places have moved to the $Stop_i^{nd}$ places (through the firing of some transition in $Tstop^{nd}$), the transition $NdtoPr$ can fire, removing all tokens from the $Stop_i^{nd}$ places and putting one token in every Run_i^{pr} place. Similarly, transition $PrtoNd$ is enabled only when all tokens have moved from the Run_i^{pr} to the $Stop_i^{pr}$ places; the firing of $PrtoNd$ brings the tokens back in each Run_i^{nd} place. Thus places Run_i^{pr} and places Run_i^{nd} cannot be simultaneously marked.

Observe that any *path* in the RG can be partitioned into (maximal) sub-paths leaving only states of the same type, so that each path can be described as an alternating sequence of non deterministic and probabilistic sub-paths. Each probabilistic sub-path can be substituted by a single "complex" probabilistic step and assigned a probability based on the weights of the transitions firing along the path. The non deterministic sub-paths can be interpreted according to different semantics (see [4] for a detailed discussion). Here we select the following semantics: a path through non deterministic states is considered as a single complex action and the only state where time is spent is the first one in the sequence (that is the state that triggers the "complex" decision multi-step). So only the first state in each path will appear as a state in the MDP (the other states in the path are *vanishing*, borrowing the terminology from the literature on generalized stochastic Petri nets).

Let us now define the reward function for a sequence of non deterministic transitions, $\sigma \in (T^{nd})^*$; abusing notation we use the same name $rt()$ for the reward function for single transitions and for transition sequences. The following definition $rt(\sigma)$ assumes that the firing order in such a sequence is irrelevant w.r.t. the reward which is consistent with an additive interpretation when several decisions are taken in one step.

Definition 8 (Transition sequence reward $rt(\sigma)$) *The reward for a non deterministic transition sequence is defined as follows:*

$$rt(\sigma) = \sum_{t \in T^{nd}} rt(t)|\sigma|_t$$

where $|\sigma|_t$ is the number of occurrences of non deterministic transition t in σ.

Generation of an MDP given a RG of a MDPN and the reward structure. The MDP can be obtained from the RG of the PN model in two steps: (1) build from the RG the RG_{nd} such that given any non deterministic state nd and any probabilistic state pr all maximal non deterministic sub-paths from nd to pr are reduced to a single non deterministic step; (2) build the RG_{MDP} (*i.e.*, a MDP) from the RG_{nd} such that given any non deterministic state nd and any probabilistic state pr, all maximal probabilistic sub-paths from pr to nd are substituted by a single probabilistic step. Finally derive the MDP reword from rs,rt and rg functions.

Let nd be a non deterministic state reached by a probabilistic transition (such states will be the non deterministic states of RG_{nd}). We focus on the subgraph "rooted" in nd and obtained by the maximal non deterministic paths starting from nd. Note that the probabilistic states occurring in this subgraph are terminal states. If there is no finite maximal non deterministic sub-paths starting from nd then no probabilistic phase can follow. So the construction is aborted. Otherwise, given every probabilistic state pr of the subgraph, one wants to obtain the optimal path $\sigma_{nd,pr}$ from nd to pr w.r.t. the reward. Once for every such pr, this path is computed, in RG_{nd} an arc is added from nd to pr labeled by $\sigma_{nd,pr}$. The arcs starting from probabilistic states are unchanged in RG_{nd}.

Thus the building of RG_{nd} depends on whether the optimization problem is a maximization or a minimization of the reward. We only explain the minimization case (the other case is similarly handled). We compute such a sequence using the Bellman and Ford (BF) algorithm for a single-source shortest paths in a weighted digraph where the transition reward is the cost function associated with the arcs. This algorithm is sound due to our (cumulative) definition for rewards of transition sequences. Note that if the BF algorithm finds a negative loop (*i.e.*, where the reward function decreases), the translation is aborted. Indeed the optimal value is then $-\infty$ and there is no optimal sequence: this problem must be solved at the design level.

We now explain how to transform RG_{nd} into the MDP RG_{MDP}. Given a probabilistic state pr and a non deterministic state nd we want to compute the probability to reach nd along probabilistic sub-paths. Furthermore, the sum of these transition probabilities over non deterministic states must be 1. So if in RG_{nd}, there is a terminal strongly connected component composed by only probabilistic states, we abort the construction. The checked condition is necessary and sufficient according to Markov chain theory. Otherwise, we obtain the transition probabilities using two auxiliary matrices. $\mathbf{P}^{(pr,pr)}$, a square matrix indexed by the probabilistic states, denotes the one-step probability transitions between these states and $\mathbf{P}^{(pr,nd)}$, a matrix whose rows are indexed by the probabilistic states and columns are indexed by non deterministic states, denotes the one-step probability transitions from probabilistic states to non deterministic ones. Let us describe how these transition probabilities are obtained. These probabilities are obtained by normalizing the weights of the transitions enabled in pr. Now again, according to Markov chain theory, matrix $\mathbf{P} = (\mathbf{Id} - \mathbf{P}^{(pr,pr)})^{-1} \circ \mathbf{P}^{(pr,nd)}$, where $\mathbf{Id}$ is the identity matrix represents the searched probabilities. A similar transformation is performed in the framework of stochastic Petri nets with immediate transitions (see [17] for the details).

Finally in the MDP, the probability distribution $p(\cdot|nd, \sigma)$ associated with state nd and (complex) action σ, assuming $nd \xrightarrow{\sigma} pr$, is given by the row vector $\mathbf{P}[pr, \cdot]$ and the reward function for every pair of state and action is defined by the following formula: $r(nd, \sigma) = rg(rs(nd), rt(\sigma))$. Since rg is not decreasing w.r.t. its second parameter, the optimal path w.r.t. rt found applying the Bellman and Ford algorithm is also optimal w.r.t. $rg(rs(nd), rt(\cdot))$.

References

[1] T. Assaf and J. B. Dugan. Diagnostic Expert Systems from Dynamic Fault Trees. In *Annual Reliability and Maintainability Symposium 2004 Proceedings*, pages 444–450, Los Angeles, CA USA, January 2004.

[2] M. Beccuti, D. Codetta-Raiteri, G. Franceschinis, and S. Haddad. A framework to design and solve Markov Decision Well-formed Net models. In *Int. Conf. on Quantitative Evaluation of Systems*, pages

165–166, Edinburgh, Scotland, UK, September 2007.

[3] M. Beccuti, G. Franceschinis, and S. Haddad. Markov Decision Petri Net and Markov Decision Well-Formed Net Formalisms. *Lecture Notes in Computer Science*, 4546:43–62, 2007.

[4] M. Beccuti, G. Franceschinis, and S. Haddad. Markov Decision Petri Net and Markov Decision Well-formed Net formalisms. Technical Report TR-INF-2007-02-01, Dipartimento di Informatica, Università del Piemonte Orientale, 2007. http://www.di.unipmn.it/Tecnical-R.

[5] A. Bobbio, G. Franceschinis, R. Gaeta, and G. Portinale. Parametric fault tree for the dependability analysis of redundant systems and its high-level Petri net semantics. *IEEE Transactions on Software Engineering*, 29(3):270–287, March 2003.

[6] H. Boudali, P. Crouzen, and M. Stoelinga. Dynamic Fault Tree Analysis Using Input/Output Interactive Markov Chains. In *Int. Conf. on Dependable Systems and Networks*, pages 708–717, June 2007.

[7] R. E. Bryant. Symbolic Boolean manipulation with Ordered Binary Decision Diagrams. *ACM Computing Surveys*, 24:293–318, 1992.

[8] I.T. Castroa and E.L. Sanjuan. An optimal repair policy for systems with a limited number of repairs. *European Journal of Operational Research*, 187(1):84–97, May 2008.

[9] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. Stochastic well-formed coloured nets for symmetric modelling applications. *IEEE Transactions on Computers*, 42(11):1343–1360, nov 1993.

[10] G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo. GreatSPN 1.7: Graphical Editor and Analyzer for Timed and Stochastic Petri Nets. *Performance Evaluation, special issue on Performance Modeling Tools*, 24(1-2):47–68, November 1995.

[11] D. Codetta-Raiteri, G. Franceschinis, and M. Gribaudo. Defining formalisms and models in the Draw-Net Modelling System. In *Int. Workshop on Modelling of Objects, Components and Agents*, pages 123–144, Turku, Finland, June 2006.

[12] D. Codetta-Raiteri, G. Franceschinis, M. Iacono, and V. Vittorini. Repairable Fault Tree for the automatic evaluation of repair policies. In *Int. Conf. on Dependable Systems and Networks*, pages 659–668, Florence, Italy, June 2004.

[13] S. Contini. *ASTRA - Advanced Software Tool for Reliability Analysis, Theoretical Manual.* European Commission Joint Research Centre (JRC), Ispra, Italy, http://www.jrc.ec.europa.eu, 1999. EUR 18727en.

[14] H. Howard. *Dynamic Programming and Markov Process*. MIT Press, 1960.

[15] G. Krishnamurthi, A. Gupta, and A. K. Somani. HIMAP: Architecture, Features, and Hierarchical Model Specification Techniques. *Lecture Notes in Computer Science*, 1469:348–351, 1998.

[16] R. Manian, D.W. Coppit, K.J. Sullivan, and J.B. Dugan. Bridging the Gap Between Systems and Dynamic Fault Tree Models. In *Annual Reliability and Maintainability Symposium*, pages 105–111, 1999.

[17] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. Wiley Series in Parallel Computing, John Wiley and Sons, 1995. Download http://www.di.unito.it/∼greatspn.

[18] M. Puterman. *Markov decision processes : Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[19] A. Rauzy. New Algorithms for Fault Trees Analysis. *Reliability Engineering and System Safety*, 05(59):203–211, 1993.

[20] R. Righter. Optimal maintenance and operation of a system with backup components. *Probability in the Engineering and Informational Sciences*, 16(3):339–349, 2002.

[21] R. A. Sahner, K. S. Trivedi, and A. Puliafito. *Performance and Reliability Analysis of Computer Systems; An Example-based Approach Using the SHARPE Software Package*. Kluwer Academic, 1996.

[22] W.G. Schneeweiss. *The Fault Tree Method*. LiLoLe Verlag, 1999.

[23] G. J. Wang and Y. L. Zhang. Optimal periodic preventive repair and replacement policy assuming geometric process repair. *IEEE Transactions on Reliability*, 55(1):118–122, March 2006.

[24] FTA-Pro tool's web page. http://www.dyadem.com/products/ftapro/index.php.

[25] Galileo tool's web page. http://www.cs.virginia.edu/∼ftree/.

[26] Relex tools' web page. http://www.relex.com/products/ftaeta.asp.

[27] Web page of the FaultTree+ tool by Isograph Software. http://www.isograph.com/faulttree.htm.

[28] Web page of the FTAnalyzer tool by Advanced Logistic Development (ALD). http://www.ald.co.il/products/FTAnalyzer.html.